# Paralelní a distribuované algoritmy - dokumentácia výpočtu úrovne vrcholu

Katarína Grešová

20. apríla 2019

## 1 Úvod

Táto dokumentácia popisuje paralelný algoritmus na výpočet úrovne vrcholov v binárnom strome – jeho implementáciu, časovú zložitosť, komunikáciu procesov a experimenty na overenie časovej zložitosti.

## 2 Rozbor a analýza algoritmu

Algoritmus výpočtu úrovne vrcholu počíta počet hrán na ceste z daného vrcholu do koreňa. Paralelná verzia tohto algoritmu využíva Eulerovú cestu a sumu suffixov na výpočet rozdielu spätných a dopredných hrán na zvyšku Eulerovej cesty od daného vrcholu do konca cesty.

Počet použitých procesorov sa odvíja od počtu zadaných vrcholov a je potrebné ho vypočítať pred spustením samotného algoritmu. Algoritmus pracuje s hranami, takže počet procesorov bude odpovedať počtu dopredných a spätných hrán zadaného stromu. Keďže pracujeme s úplným binárnym stromom, ktorý má $n$ vrcholov, budeme potrebovať $2 * (n - 1)$ procesorov.

Algoritmus má tri hlavné časti: priradenie váh hranám, spočítanie Eulerovej cesty a spočítanie sumy suffixov. Dopredným hranám je priradená váha -1 a spätným hranám váha 1. To, či je hrana dopredná alebo spätná, je určené číslom procesoru, ktorý sa o ňu stará. Hrany s číslom v dolnej polovici hodnôt sú dopredné a hrany v hornej polovici hodnôt sú spätné. Váhy sú uložené do poľa hrán a rozdistribuované medzi všetky procesory pomocou funkcie MPI_Allgather[1].

Spočítanie Eulerovej cesty spočíva v tom, že každá hrana zistí svojho následníka. Zistenie následníkov sa deje pomocou funkcií, ktoré z čísla procesoru aktuálnej hrany a zo znalosti číslovania hrán vypočítajú číslo procesoru, ktoré prislúcha nasledujúcej hrane. Na záver je ešte ošetrený následník poslednej hrany Eulerovej cesty. Následník tejto hrany je nastavený na seba. Tým sa rozpojú Eulerova kružnica a vznikne Eulerova cesta, ktorá má cyklus na poslednom prvku, čo sa hodí pri paralelnom výpočte sumy suffixov. Následníci hrán sú uložení do poľa následníkov rozdistribuovaní medzi všetky procesory pomocou funkcie MPI_Allgather.

Ďalším krokom je paralelný výpočet sumy suffixov. Na vstupe je pole váh jednotlivých hrán a pole následníkov jednotlivých hrán. Na začiatku sú sumy jednotlivých hrán inicializované na ich prislúchajúcu váhu. Okrem poslednej hrany, ktorej suma je inicializovaná na nulu (neutrálny prvok k operácií sčítania). Nasleduje cyklus, kde v každej iterácií, každý procesor k svojej sume pripočíta sumu svojho následníka a aktualizuje hodnotu svojho následníka. Aktuálne hodnoty v poli súm a v poli následníkov sú rozdistribuované medzi všetky procesory v každej iterácií potomocu funkcie MPI_Allgather. Po ukončení cyklu sú ešte hodnoty sumy suffixov upravené o pôvodnú váhu poslednej hrany, ktorá bola počas výpočtu nastaevná na nulu.

Na záver algoritmu, procesor s hodnotou 0 zapíše výsledok na výstup. Úrovne vrcholov sa získajú zo sumy suffixov dopredných hrán, ktoré do nich smerujú, upravené ohodnotu +1. Úroveň koreňového vrcholu je pevne 0. Úrovne nasledujúcich vrcholov sú získavané postupne z poľa sumy suffixov.

[1] https://www.mpich.org/static/docs/latest/www3/MPI_Allgather.html

## 3 Teoretická zložitosť

Na výpočet časovej zložitosti budeme analyzovať nasledujúce kroky:

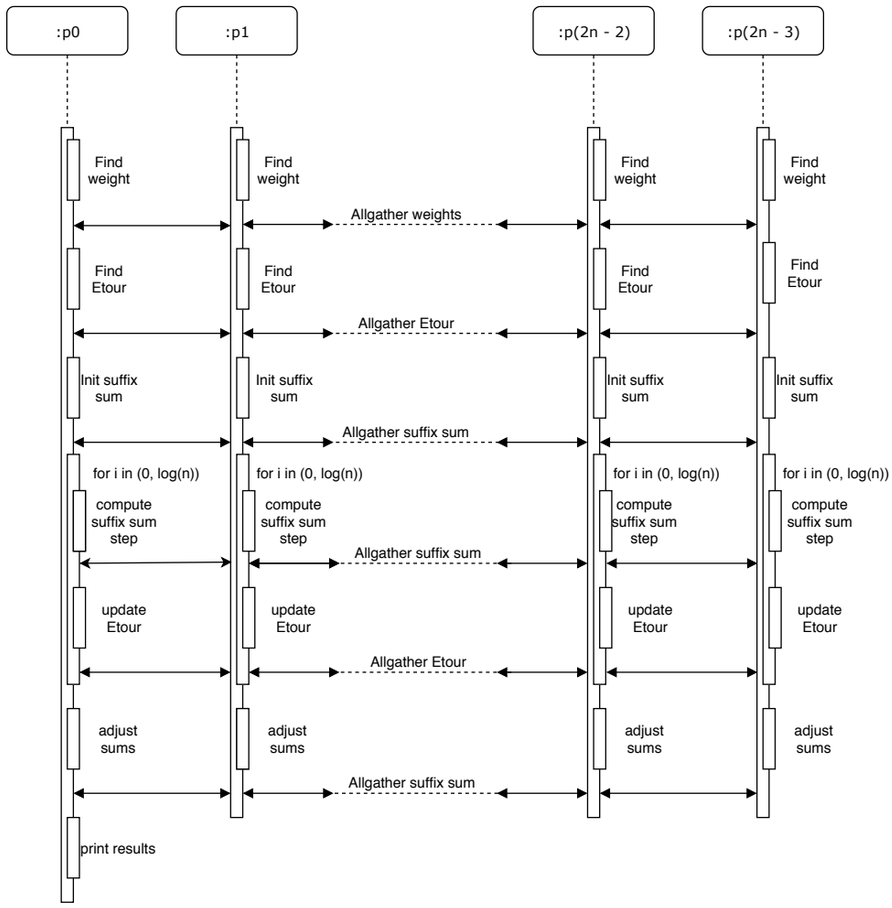
1. Priradenie váh hranám – $O(c)$
2. Spočítanie Eulerovej cesty – $O(c)$
3. Spočítanie sumy suffixov – $O(\log_2 n)$

Celkovo:

- $t(n) = O(\log_2 n)$
- $p(n) = 2 * n - 2 = O(n)$
- $c(n) = O(n * \log_2 n)$

## 4 Komunikačný protokol

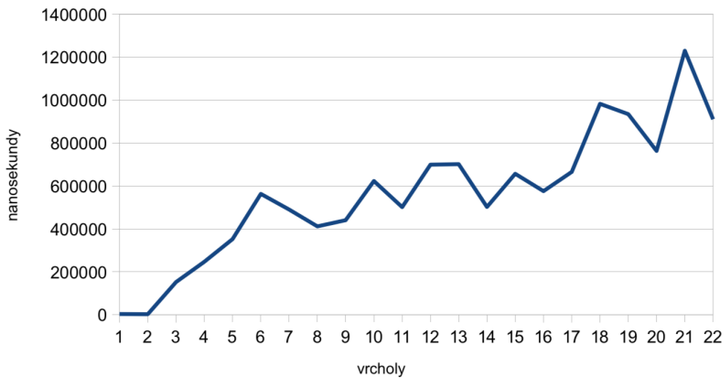
Na obrázku 1 je znázornený komunikačný protokol medzi $2(n-1)$ procesormi.




Obr. 1: Komunikačný protokol v aplikácii

## 5 Experimenty a testovanie

Testovanie časovej zložitosti bolo vykonané so vstupnými sekvenciami od dĺžky 1 po dĺžku 22. Štatistická relevancia bola zaistená spustením algoritmu s každou dĺžkou vstupu 50-krát a odstránením jedného najkratšieho a jedného najdlhšieho behu. Výsledky meraní sú znázornené v grafe 2.




Obr. 2: Výsledky merania časovej zložitosti algoritmu

## 6 Záver

Aj napriek násobnému meraniu časovej zložitosti výsledky mierne kolísajú, čo môže byť pripísané nerovnomernému vyťaženiu serveru merlin, kde bolo meranie vykonané a nedostatočnému počtu meraní. Namerané dáta však vykazujú mierne logaritmické závislosť, ktorá odpovedá odvodenej teoretickej zložitosti.